Assignment #4:
Due: October 15
Proofs or counter-examples absolutely necessary for this assignment!

1. Let $G = [V,E]$ be an undirected graph whose edges are not known in advance. We want to check if it is connected. The only questions that we are allowed to ask are of the form: "Is there an edge between vertices $i$ and $j$?". Using an adversary argument show that any correct deterministic algorithm to decide if $G$ is connected must ask $\Omega(n^2)$ questions.

2. Consider the activity selection problem discussed in 16-1 of your book. For each of the following greedy algorithms, either prove that the algorithm solves the problem correctly or give a counter-example to show that it does not always work correctly:

   (a) Select the activity that overlaps with the least number of other activities among those that are compatible with previously selected activities at each step of the algorithm.

   (b) Select the activity with the latest starting time among those that are compatible with previously selected activities at each step of the algorithm.

3. There are $n$ objects of each of $m$ colors. Let $w_{i,j}$ be the weight of $i^{th}$ object with $j^{th}$ color for $i = 1, 2, ..., n; j = 1, 2, ..., m$. Let $b_j$ be the maximum number of objects of $j^{th}$ color we can select for $j = 1, 2, ..., m$. Show how to select the set of maximum total weight by describing a valid greedy algorithm. Justify your answer.

4. The following problem is known in the literature as the knapsack problem: We are given $n$ objects each of which has a weight and a value. Suppose that the weight of object $i$ is $w_i$ and its value is $v_i$. We have a knapsack that can accommodate a total weight of $W$. We want to select a subset of the items that yields the maximum total value without exceeding the total weight limit.

   (i) If all $v_i$ are equal, what would the greedy algorithm yield? Is this optimal?

   (ii) If all $w_i$ are equal, what would the greedy algorithm yield? Is this optimal?

   (iii) How should the greedy algorithm be designed in the general case? Is this optimal? [Be careful to distinguish between two versions of the problem: in one we are allowed to select fractional items and in the other we are not allowed to do this.]
5. Consider the following generalization of a scheduling example done in class:
   We have \( n \) customers to serve and \( m \) identical machines that can be used for this (such as tellers in a bank). The service time required by each customer is known in advance: customer \( i \) will require \( t_i \) time units \( (1 \leq i \leq n) \). We want to minimize \( \sum_{i=1}^{n} C_i(S) \), where \( C_i(S) \) represents the time at which customer \( i \) completes service in schedule \( S \). How should the greedy algorithm work in this case? Is it guaranteed to produce optimal solutions?

6. Challenge Problem I: Show that for any binary tree with \( k \) leaf nodes, the sum of the depth of the leaf nodes is at least \( k \lg k \). Using this show that the average time complexity of any sorting algorithm based on comparisons is \( \Omega(n \lg n) \).

7. Challenge Problem II: Show that the lower bound for problem #1 is \( \frac{n(n^2-1)}{2} \) [i.e. we must check for each edge before deciding if \( G \) is connected or not].

8. Challenge Problem III: Input: A set \( V \) of size \( n \); a vector \( [a_1, a_2, ..., a_n] \) of positive integers. We want to construct a tree whose nodes are \( V \); for node \( i \) the number of its children in the tree must be no more than \( a_i \) for \( i = 1, 2, ..., n \). You are permitted to make any node the root of the tree. We want to minimize the height of the tree. Develop a greedy algorithm that produces the desired tree and prove that your results are correct.